(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number : **0 498 654 A2**

(12) EUROPEAN PATENT APPLICATION

(21) Application number : **92301024.3**

(51) Int. Cl.[5] : **G06F 9/38**

(22) Date of filing : **06.02.92**

(30) Priority : **08.02.91 JP 16716/91**
13.02.91 JP 19849/91

(43) Date of publication of application :
12.08.92 Bulletin 92/33

(84) Designated Contracting States :
DE FR GB

(71) Applicant : **FUJITSU LIMITED**
1015, Kamikodanaka Nakahara-ku
Kawasaki-shi Kanagawa 211 (JP)

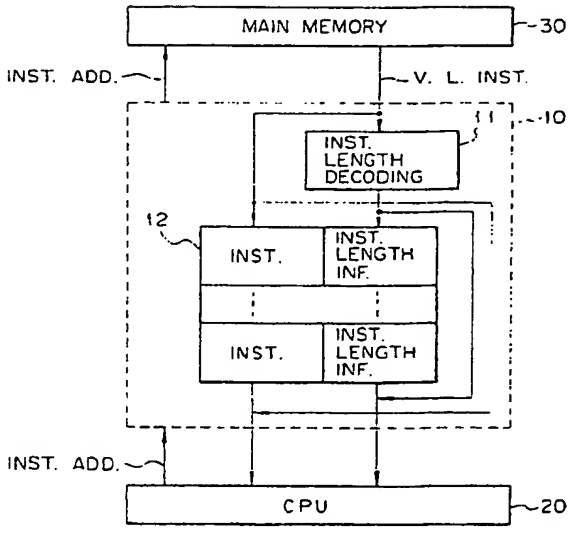
(72) Inventor : **Sato Taizo, c/o Fujitsu Limited**
1015, Kamikodanaka, Nakahara-ku
Kawasaki-shi, Kanagawa 211 (JP)
Inventor : **Fujihara, Atsushi, c/o Fujitsu Limited**
1015, Kamikodanaka, Nakahara-ku
Kawasaki-shi, Kanagawa 211 (JP)

(74) Representative : **Billington, Lawrence Emlyn et al**
HASELTINE LAKE & CO Hazlitt House 28
Southampton Buildings Chancery Lane
London WC2A 1AT (GB)

(54) Cache memory processing instruction data and data processor including the same.

(57) A cache memory (10) for processing at least one variable length instruction taken from a memory (30) and feeding the processed information to a control unit (20), and a data processor including the cache memory. The cache memory includes a unit (11) for decoding an instruction length of the variable length instruction taken from the memory, and a unit (12) for storing the variable length instruction from the memory, together with the decoded instruction length information. The variable length instruction and the instruction length information thereof are fed to the control unit. Accordingly, the cache memory enables the control unit to simultaneously decode a plurality of variable length instructions and thus realize a high speed computing processing.

Fig. 4



EP 0 498 654 A2

The present invention relates to a cache memory connected between a central processing unit (CPU) executing a high speed computing processing and a main memory effecting a relatively low speed operation in a computer. More particularly, it relates to a constitution of a cache memory which processes instruction data from the main memory and thus is adapted to realize a high speed decoding of the instruction data in the CPU.

Recently, CPUs which execute a greater part of an instruction in one cycle have been developed and, nevertheless, they are demanded to execute a further higher speed computing processing. To this end, cache memories connected to the respective CPUs are also demanded to effect a high speed operation accordingly.

As one approach to realize a high speed computing processing, an idea of increasing a frequency of the operational clock can be proposed. This, however, is difficult to realize because a peripheral circuit, for example, constituted by transistor-transistor logics, cannot satisfactorily meet the requirements for the frequency. Accordingly, other approaches to realize the high speed computing processing without increasing the frequency of the operational clock have been demanded.

As an approach to meet the demand, a technique of simultaneously executing a plurality of instructions is known, which is roughly classified into two categories. One is the case that each instruction has a fixed length, and the other is the case that each instruction has a variable length. With respect to a data processor which processes a plurality of fixed length instructions, a position of a next instruction succeeding a certain instruction is fixed and, accordingly, it is possible to simultaneously execute the two neighbouring instructions by decoding them.

However, with respect to a data processor which processes a plurality of variable length instructions as in The Real time Operating system Nucleus (TRON) specification, a problem is posed. Namely, since the position of the succeeding instruction is changed or not fixed, it is impossible to simultaneously execute the two neighbouring instructions so long as the position of the succeeding instruction is not specified by any means.

Thus, in a prior art data processor processing variable length instructions as in TRON specification, a problem occurs in that it is impossible to simultaneously execute a plurality of instructions and thus realize a high speed computing processing. In other words, to realize the high speed computing processing, a cache memory connected between a CPU and a main memory must be improved with respect to the processing of instruction data from the main memory. To cope with this, various approaches are proposed and, however, a known effective improvement has not been proposed.

Note, the problems in the prior art will be explained later in detail in contrast with the preferred embodiments of the present invention.

An object of the present invention is to provide a cache memory adapted to enable a CPU to simultaneously decode a plurality of instructions even if they are variable length ones, and thus realize a high speed computing processing.

According to a first aspect of the present invention, there is provided a cache memory for storing at least one variable length instruction taken from a memory and feeding the stored information to a control unit, the cache memory including: an instruction length decoding unit for decoding an instruction length of the variable length instruction taken from the memory; and an instruction storing unit for storing the variable length instruction from the memory, together with an instruction length information obtained by the instruction length decoding unit, wherein the variable length instruction and the instruction length information thereof are fed to the control unit.

Also, according to a second aspect of the present invention, there is provided a cache memory for storing an instruction data taken from a memory and feeding the stored information to a control unit, the cache memory including: a unit for generating a predecoded information as an auxiliary of a decoding of the instruction data; a memory unit for storing the predecoded information together with the instruction data; a unit for checking a predecoded information output from the memory unit together with a corresponding instruction data in the cache hit state; a rewriting unit, where the checked predecoded information proves to be invalid or wrong, for rewriting the corresponding predecoded information in the memory unit to a right one; and a control information outputting unit, where the checked predecoded information proves to be invalid or wrong, for outputting a control information reflecting a result of the check to the control unit.

Other objects and features of the present invention will be described hereinafter in detail by way of preferred embodiments with reference to the accompanying drawings, in which:

Fig. 1 is a block diagram showing a constitution of a first prior art data processor using a cache memory;

Fig. 2 is a block diagram showing a constitution of a second prior art data processor using a cache memory;

Fig. 3 is a block diagram showing a constitution of a third prior art data processor using a cache memory;

Fig. 4 is a block diagram showing a fundamental constitution of the cache memory according to a first aspect of the present invention;

Fig. 5 is a block diagram showing a constitution of an embodiment according to the first aspect of the present invention;

Fig. 6 is a block diagram showing a constitution of the instruction predecoder shown in Fig. 5;

Fig. 7 is a block diagram schematically showing a constitution of the data memory shown in Fig. 5;

Figs. 8A and 8B are block diagrams showing a constitution by which the CPU can simultaneously process two instructions with or without extended parts;

Figs. 9A to 9C are block diagrams showing another constitution by which the CPU can simultaneously process two instructions each having a variable length extended part;

Fig. 10 is a block diagram showing a constitution of a fourth prior art data processor using a cache memory;

Fig. 11 is a block diagram showing a fundamental constitution of the cache memory according to a second aspect of the present invention; and

Fig. 12 is a block diagram showing a constitution of an embodiment according to the second aspect of the present invention.

Throughout the description, identical references used in connection with the drawings indicate like constituent elements and thus the repetition of explanation thereof is omitted.

A preferred embodiment according to the first aspect of the present invention will be explained with reference to Figs. 1 to 9C.

First, for better understanding of the preferred embodiment, the related prior art will be explained with reference to Figs. 1 to 3.

Figure 1 illustrates a constitution of a first prior art data processor using a cache memory.

In the illustration, reference 1a denotes a cache memory connected between a CPU 2 and a main memory 3 in the data processor. The cache memory 1a includes an instruction storing unit 4 for storing instructions taken from the main memory 3, and a cache hit/miss judging unit 5 for comparing an instruction address output from the CPU 2 with a comparison address retained in the cache memory 1a and judging whether a cache "hit" state or a cache "miss" state is encountered.

In the data processor thus constituted, where the CPU 2 fetches an instruction, it first makes an address access to the cache memory 1a. Where the instruction to be fetched is present in the cache memory 1a (i.e., cache "hit" state), it is taken into the CPU 2. On the other hand, where the instruction to be fetched is not present in the cache memory 1a (i.e., cache "miss" state), an address access is made to the main memory 3 and thus the corresponding instruction is read out. The instruction is taken into the CPU 2 and stored in the instruction storing unit 4 of the cache memory 1a.

In the above constitution, it is assumed that the CPU 2 is constituted so as to process a plurality of instructions. Where each instruction has a fixed length, the position of an instruction succeeding a certain instruction is fixed and, accordingly, the CPU 2 can simultaneously execute the plurality of instructions by decoding them.

Where each instruction has a variable length, however, the position of the succeeding instruction is changed. Accordingly, so long as the position of the succeeding instruction is not specified by any means, the CPU 2 cannot simultaneously execute the plurality of instructions.

Figure 2 illustrates a constitution of a second prior art data processor using a cache memory.

The illustrated data processor is directed to a simultaneous execution of a plurality of variable length instructions and includes a cache memory 1 which receives and transmits instruction addresses and variable length instructions between a CPU 2a and the main memory 3. The CPU 2a includes an instruction register 8 for temporarily storing instruction data from the cache memory 1, an instruction predecoder 6 for predecoding the instruction data from the instruction register 8 to obtain an instruction length thereof, an instruction register 9 for storing the instruction data from the instruction register 8 for a predetermined time (corresponding to the time required for the predecoding), and an instruction decoder 7 for decoding the instruction data from the instruction register 9 and the predecoded information (i.e., instruction length information) from the instruction predecoder 6.

According to the constitution, it is possible to simultaneously supply the instruction decoder 7 with two instructions, using the instruction predecoder 6 and the instruction register 9, and thus effect a simultaneous execution of the two instructions.

The constitution, however, has a disadvantage in that it takes time to predecode the instruction data. Also, since the instruction data has a variable length, it is not easy for the instruction predecoder 6 to simultaneously predecode two variable length instructions. As a result, a problem occurs in that it is impossible to successively feed instructions to the instruction decoder 7 and thus the computing processing speed is lowered.

Figure 3 illustrates a constitution of a third prior art data processor using a cache memory.

In the illustration, reference 1b denotes a cache memory which receives and transmits instruction addresses and variable length instructions between the CPU 2 and the main memory 3. The cache memory 1b includes an instruction register 8a for storing a variable length instruction taken from the main memory 3, an instruction decoder 7a for decoding the variable length instruction from the instruction register 8a, and an instruction storing unit 4a for storing the decoded information from the instruction decoder 7a.

This constitution, however, has a disadvantage in that a data size of the instruction storing unit 4a must be selected to be as large as that of data having the longest instruction length. Thus, the constitution in which the cache memory 1b is provided with the instruction decoder 7a is not suitable in the practical use or application.

Figure 4 illustrates a fundamental constitution of the cache memory according to the first aspect of the present invention.

In the illustration, reference 10 denotes a cache memory which processes at least one variable length instruction taken from an external memory (e.g., main memory 30) and feeds the processed information to an external control unit (e.g., CPU 20). The cache memory 10 includes an instruction length decoding unit 11 for decoding an instruction length of the variable length instruction taken from the external memory, and an instruction storing unit 12 for storing the variable length instruction from the external memory, together with an instruction length information obtained by the instruction length decoding unit, wherein the variable length instruction and the instruction length information thereof are fed to the external control unit.

In the above constitution, where the cache memory 10 encounters the cache "hit" state, the corresponding variable length instruction and the instruction length information thereof are read out from the instruction storing unit 12 and fed to the CPU 20. Thus, the CPU 20 can simultaneously decode a plurality of variable length instructions. This contributes to a high speed computing processing.

On the other hand, where the cache memory 10 encounters the cache "miss" state, the corresponding variable length instruction is taken from the main memory 30 into the cache memory 10 and the instruction length thereof is decoded by the instruction length decoding unit 11. Then, the variable length instruction concerned and the instruction length information thereof are fed to the CPU 20. Accordingly, time required until the variable length instruction concerned is decoded by the CPU 20 becomes longer than that in the cache "hit" state. In this case (cache "miss" state) too, however, it is of course possible to simultaneously decode a plurality of variable length instructions in the CPU 20.

Figure 5 illustrates a constitution of the preferred embodiment according to the first aspect of the present invention.

In the illustration, reference 100 denotes a cache memory which is integrated into a monolithic microprocessor MP including the CPU 20. Namely, the cache memory 100 is formed together with the CPU 20 in a single semiconductor substrate (not shown).

In the cache memory 100, reference 115 denotes an instruction address inputting unit for receiving an address from the CPU 20, the address corresponding to an instruction to be fetched from the cache memory 100 or the main memory 30.

Reference 116 denotes a cache hit/miss judging unit which compares an instruction address output from the CPU 20 with a comparison address (tag) retained in the unit 116 and, based on a result of the comparison, judges whether a cache "hit" state or a cache "miss" state is encountered. The cache hit/miss judging unit 116 includes a tag memory 117 constituted by a random access memory (RAM) and retaining the comparison address, and an address comparing unit 118 for comparing the instruction address with the comparison address.

Reference 119 denotes an instruction address outputting unit which outputs the above instruction address from the instruction address inputting unit 115 to the memory 30 when it is informed of the cache "miss" state by the address comparing unit 118.

Also, reference 120 denotes an instruction data inputting unit for receiving at least one variable length instruction output from the main memory 30.

Reference 121 denotes an instruction predecoder characteristic of the present embodiment. The instruction predecoder 121 decodes the variable length instruction taken from the main memory 30, prior to a decoding by an instruction decoder (described later) provided in the CPU 20, and produces an instruction length information thereof. The instruction length information is produced, for example, as shown in Table below.

Table

inst. length information (or predecoded result)	contents
0 0 0	NOT predecoded
1 0 0	INTERMEDIATE code
1 0 1	END of inst.
1 1 0	HEAD of inst.
1 1 1	HEAD/END of inst.

Reference 122 denotes an instruction storing unit for storing instructions including variable length instructions and the corresponding instruction length information. Also, reference 123 denotes a bypassing unit and reference 124 denotes an instruction data outputting unit. The instruction storing unit 122 includes an input/output (I/O) buffer 125 and a data memory 126 constituted by a RAM and storing the instruction received at the instruction data inputting unit 120 and the corresponding instruction length information obtained by the instruction predecoder 121.

Where the I/O buffer 125 is informed of the cache "hit" state by the address comparing unit 118, it reads the corresponding instruction and the instruction length information thereof from the data memory 126 and transmits the read data to the instruction data outputting unit 124. On the other hand, where the I/O buffer 125 is informed of the cache "miss" state by the address comparing unit 118, it receives the corresponding instruction data from the instruction data inputting unit 120 and the instruction length information thereof from the instruction predecoder 121 and writes the received data into the data memory 126.

Also, where the bypassing unit 123 is informed of the cache "miss" state by the address comparing unit 118, it receives the corresponding variable length instruction from the instruction data inputting unit 120 and the instruction length information thereof from the instruction predecoder 121 and transmits the received data to the instruction data outputting unit 124.

The instruction data outputting unit 124 receives the instruction (or variable length instruction) and the corresponding instruction length information from the instruction storing unit 122 or the bypassing unit 123 and outputs the received data to the CPU 20.

Figure 6 illustrates a constitution of the instruction predecoder 121.

In the present embodiment, a variable length instruction data is transmitted from the instruction data inputting unit 120 via a bus having a data width of sixty-four bits and divided into four data blocks to be input to the instruction predecoder 121. References 211 to 214 denote instruction latch circuits each for latching the corresponding data of sixteen bits (one half word [1 HW]) in the respective data blocks. References 221 to 224 denote format decoders each for decoding a format of the data of sixteen bits from the corresponding instruction latch circuit, and references 231 to 234 denote effective address decoders each for decoding an effective address based on eight bits among the corresponding data of sixteen bits.

Also, references 241 to 244 denote instruction length determining circuits each for determining an instruction length based on decoded results of the corresponding format decoder and effective address decoder. In this case, each of the instruction length determining circuits 241 to 243 is controlled by an instruction head indicating circuit 260 and thus controls the instruction length determining circuit or circuits in the lower rank side. For example, the instruction length determining circuit 243 is controlled by the instruction head indicating circuit 260 and the instruction length determining circuits 241, 242 and thus controls the instruction length determining circuit 244. The instruction length determining circuit 244 is controlled by the instruction length determining circuits 241 to 243 in the higher rank side.

Reference 250 denotes an output pattern generating circuit which generates an instruction length information for the input variable length instruction data based on each output of the instruction length determining circuits 241 to 244. The output pattern generating circuit 250 also generates a control signal for indicating which block of the above four data blocks is at the head of the variable length instruction. The instruction head indicating circuit 260 responds to the control signal and controls the instruction length determining circuits 241 to 243 to be enable or disable.

Figure 7 schematically illustrates a constitution of the data memory 126.

As illustrated in Fig. 7, the data memory 126 is divided into four blocks B1 to B4, each block containing instruction data of sixteen bits (1 HW) and the corresponding instruction length information of three bits. Namely,

the instruction data (or variable length instruction data) is comprised of sixty-four bits at maximum.

In the above constitution of the cache memory according to the present embodiment, a variable length instruction taken from the main memory 30 is stored in the instruction storing unit 122, together with the corresponding instruction length information decoded by the instruction predecoder 121.

Therefore, where the cache memory 100 encounters the cache "hit" state, the corresponding variable length instruction and the instruction length information thereof are read out from the instruction storing unit 122 and fed via the instruction data outputting unit 124 to the CPU 20. Thus, the CPU 20 can simultaneously decode a plurality of variable length instructions as described later. This contributes to a high speed computing processing.

Also, where the cache memory 100 encounters the cache "miss" state, the corresponding variable length instruction is taken from the main memory 30 into the cache memory 100 and the instruction length thereof is decoded by the instruction predecoder 121. Then, the variable length instruction concerned and the corresponding instruction length information are fed to the CPU 20. Accordingly, time required until the variable length instruction concerned is decoded by the CPU 20 becomes longer than that in the cache "hit" state. In this case too, however, it is of course possible to simultaneously decode a plurality of variable length instructions in the CPU 20.

Figures 8A and 8B illustrate a constitution by which the CPU 20 can simultaneously process two instructions with or without extended parts.

In the illustration, reference "C" denotes a train of instruction codes (i.e., a single instruction) constituted by one set consisting of a basic part and an extended part or a combination of a plurality of sets. The basic part includes a code for classifying the kind of the instruction, and the extended part is added by the designation of the basic part and has a variable length. Reference 301 denotes a cache memory which may be the cache memory 100 shown in Fig. 5. Reference 315 denotes a mark appending unit constituted by, e.g., a predecoder. The mark appending unit 315 judges whether data of every unit length in the train of instruction codes "C" is a basic part or not, and appends a predetermined mark "M" thereto.

Reference 302 denotes an instruction buffer unit, which includes a code memory unit 303 for reading the train of instruction codes "C" and storing them, a mark memory unit 316 for storing the mark "M", a write pointer 306 for designating a write address in the code memory unit 303, a read pointer 307 for designating a head position of a predetermined train of instruction codes to be selectively output from the code memory unit 303, an output position indicating circuit 317, responsive to the output of the read pointer 307 and information from the mark memory unit 316, for designating an output position of the predetermined train of instruction codes, and an output selecting circuit 308, responsive to the output of the output position indicating circuit 317 and the information from the mark memory unit 316, for selectively outputting the predetermined train of instruction codes from the code memory unit 303.

Reference 309 denotes an instruction decoder constituted by a programmable logic array (PLA) and reference 310 denotes an instruction register. The instruction decoder 309 and the instruction register 310 are connected via a bus 314 to the output selecting circuit 308. The bus 314 is sectioned into a plurality of different fields "X", "Y".

The output selecting circuit 308 and the output position indicating circuit 317 constitute a control unit 320. As a whole, the control unit 320 links specific portions at which the mark "M" indicates a basic part in the predetermined train of instruction codes, within a predetermined range, and controls the linked data to be output to one field of the bus 314. In the like manner, the control unit 320 controls portions at which the mark "M" indicates an extended part in the predetermined train of instruction codes, to be output to another field of the bus 314.

The train of instruction codes output to the bus 314 is input via the instruction register 310 to an immediate and displacement generating circuit 312. The circuit 312 executes a predetermined computing based on information from the instruction decoder 309 and outputs the executed result to an instruction executing unit 313. Additionally, reference 311 denotes a micro ROM for controlling micro instructions to be executed by the instruction executing unit 313.

Figures 9A to 9C illustrate another constitution by which the CPU 20 can simultaneously process two instructions each having a variable length extended part.

The fundamental constitution and the operation thereof are substantially the same as those in Figs. 8A and 8B, except that there are provided a pair of output selecting circuits 308a, 308b, a pair of instruction decoders 309a, 309b, a pair of instruction registers 310a, 310b, and a pair of instruction executing units 313a, 313b. Note, each instruction executing unit 313a, 313b directly executes the respective instruction based on information from the corresponding instruction decoder 309a, 309b and the output of the corresponding instruction register 310a, 310b.

The control mark "M" requires at least two bits per unit code length for the control of the instruction buffer

unit 302a. For example, "00" indicates portions except for a basic part; "01" indicates a basic part decodable simultaneously with a previous basic part; "10" indicates a basic part in the same instruction as a previous basic part, but not decodable simultaneously therewith; and "11" indicates a basic part in the instruction different from a previous basic part.

Since the pair of instruction decoders 309a, 309b are provided, the greatest length of an instruction code to be processed in one cycle becomes twice that in the case of one instruction decoder. To cope with this disadvantage, the number of the read port is increased to twice that in the case of one instruction decoder.

Also, the output selecting circuits 308a and 308b have different constitutions. Namely, the train of instruction codes to be sent to the instruction decoder 309a is selected from the output of the first read port, while the train of instruction codes to be sent to the instruction decoder 309b lies within the first read port or within both the first read port and the second read port. To this end, the second output selecting circuit 308b must select the output to be sent to the bus from among all of the outputs of the first and second read ports.

Next, a preferred embodiment according to the second aspect of the present invention will be explained with reference to Figs. 10 to 12 and Fig. 2.

In the like manner as the first embodiment, the related prior art will be first explained with reference to Figs. 10 and 2. Note, the explanation on Fig. 2 is previously given and thus is omitted.

Figure 10 illustrates a constitution of a fourth prior art data processor using a cache memory.

In the illustration, reference 1c denotes a cache memory which receives and transmits instruction data and instruction addresses between the CPU 2 and the main memory 3. The cache memory 1c includes an instruction address inputting unit 41 for receiving an instruction address from the CPU 2, an instruction address outputting unit 42 for outputting the instruction address to the main memory 3 in the cache "hit" state, an instruction data inputting unit 43 for receiving a corresponding instruction data from the main memory 3, an instruction data outputting unit 44 for outputting the instruction data and the corresponding predecoded information (described later) to the CPU 2 under a predetermined condition, an address comparing unit 45 for comparing the instruction address from the instruction address inputting unit 41 with a comparison address and judging whether a cache "hit" state or a cache "miss" state is encountered, an instruction predecoder 46 for predecoding the instruction data from the instruction data inputting unit 43, and a data memory unit 47 for storing the predecoded information together with the corresponding instruction data and, based on a result of the judging of the address comparing unit 45, outputting the instruction data and the predecoded information to the instruction data outputting unit 44.

According to the constitution, since the predecoded information is stored in the data memory unit 47 together with the corresponding instruction data, the instruction data and the predecoded information can be simultaneously fed to the CPU 2 in the cache "hit" state. Namely, the CPU 2 can simultaneously decode two instructions.

A problem is posed, however, where a loading is carried out with units of every data block by a wrap-around in the registration of the instruction data into the data memory unit 47. Namely, since it is impossible to discriminate whether a head portion of the respective data block is a break of the instruction, it is impossible to usually store a valid predecoded information. Such a problem occurs in the case that access is started from midway the data block, e.g., based on a branch processing. Also, where the predecoded information proves to be "invalid" even in the cache "hit" state, re-predecoding of an instruction data must be carried out. This, however, leads to a cancellation of the above advantage (simultaneous decoding of two instructions) and thus is not preferable.

Figure 11 illustrates a fundamental constitution of the cache memory according to the second aspect of the present invention.

In the illustration, reference 10a (or 10b) denotes a cache memory which processes an instruction data "A" taken from an external memory (e.g., main memory 30) and feeds the processed information to an external control unit (e.g., CPU 20). The cache memory 10a includes: a unit 14 for generating a predecoded information "B" as an auxiliary of a decoding of the instruction data; a memory unit 15 for storing the predecoded information together with the instruction data; a unit 16 for checking a predecoded information output from the memory unit together with a corresponding instruction data in the cache hit state; a rewriting unit 17, where the checked predecoded information proves to be invalid or wrong, for rewriting the corresponding predecoded information in the memory unit to a right one; and a control information outputting unit 18,18a or 18b, where the checked predecoded information proves to be invalid or wrong, for outputting a control information reflecting a result of the check to the external control unit.

The control information outputting unit may be constituted by a unit 18, where the checked predecoded information proves to be invalid or wrong, for informing the CPU 20 of the invalidity of the predecoded information. Otherwise, the control information outputting unit may be constituted by a unit 18a, where the checked predecoded information proves to be invalid or wrong, for outputting a re-predecoded information to the CPU

20, or may be constituted by a unit 18b, where the checked predecoded information proves to be invalid or wrong, for rewriting the corresponding predecoded information to be fed to the CPU 20 to a right one.

Also, in a modification of the present invention, the cache memory 10b may include a unit 19 having both the function of generating the predecoded information "B" and the function of checking a predecoded information output from the memory unit 15 together with a corresponding instruction data in the cache hit state.

According to the second aspect of the present invention, the instruction data "A" together with the predecoded information "B" is stored in the memory unit 15 and, where the cache memory encounters the cache "hit" state, the instruction data and the corresponding predecoded information are read out from the memory unit 15 and fed to the CPU 20. Thus, the CPU 20 can utilize the predecoded information without an overhead in the instruction predecoding and thus simultaneously decode a plurality of instructions. This contributes to a high speed of the computing processing.

Also, the checking unit 16 checks a validity or invalidity of the predecoded information output from the memory unit 15 and, where the predecoded information proves to be invalid, the rewriting unit 17 rewrites the invalid predecoded information to a right one. As a result, it is possible to remove an overhead in the next (second) instruction fetch and thereafter. Therefore, even when a loading is carried out with units of every data block by a wrap-around in the registration of the instruction data into the memory unit 15, it is possible to usually store a valid predecoded information. In other words, it is possible to prevent a lowering in the performance which may possibly occur where a non-predecoded information is registered into the memory unit 15. This contributes to an improvement in the performance of the CPU 20.

Additionally, where the CPU 20 processes a variable length instruction, an instruction length information corresponding to the variable length instruction is substituted for the above predecoded information and processed in the like manner.

Figure 12 illustrates a constitution of the preferred embodiment according to the second aspect of the present invention.

The illustrated constitution and the operation thereof are substantially the same as those in Fig. 5, except a number of points described below, and thus the explanation thereof is omitted.

First, the cache memory 100a of the present embodiment processes the predecoded information in place of the instruction length information used in the cache memory 100 of Fig. 5. Where the CPU 20 processes a variable length instruction, however, the predecoded information can be substituted by an instruction length information corresponding to the variable length instruction.

When the cache "miss" state is encountered, the instruction predecoder 121a predecodes an instruction data from the instruction data inputting unit 120. Also, when the cache "hit" state is encountered, the instruction predecoder 121a checks whether a predecoded information from the I/O buffer 125a is a right one for the corresponding instruction data, and outputs the checked result (predecoded result).

The data memory 126a is constituted by a RAM and stores the instruction data received at the instruction data inputting unit 120 and the corresponding predecoded information.

Where the I/O buffer 125a is informed of the cache "hit" state by the address comparing unit 118, it reads the corresponding instruction data and the predecoded information thereof from the data memory 126a and transmits the read data to the instruction data outputting unit 124a. On the other hand, where the I/O buffer 125a is informed of the cache "miss" state by the address comparing unit 118, it receives the corresponding instruction data from the instruction data inputting unit 120 and the predecoded result thereof from the instruction predecoder 121a and writes the received data into a designated address in the data memory 126a. Also, the I/O buffer 125a has the function of rewriting the predecoded information written in the designated address of the data memory 126a to a right one based on the rewrite instruction from the instruction predecoder 121a.

When the cache "miss" state is encountered, the instruction data outputting unit 124a receives the corresponding instruction data from the instruction data inputting unit 120 and the predecoded result thereof from the instruction predecoder 121a and transmits the received data to the CPU 20. Also, where the cache "hit" state is encountered, the instruction data outputting unit 124a receives the corresponding instruction data and the predecoded information thereof from the I/O buffer 125a and transmits the received data to the CPU 20. Furthermore, where the predecoded result from the instruction predecoder 121a indicates "invalid", the instruction data outputting unit 124a informs the CPU 20 of the invalidity of the predecoded information and transmits the right predecoded information rewritten by the I/O buffer 125a to the CPU 20.

The predecoded result or information by the instruction predecoder 121a is shown in the aforementioned Table.

The predecoded result is produced with units of every instruction length and stored in the data memory 126a together with the corresponding instruction data. Also, the predecoded result indicates that an instruction starting with an instruction data indicating the head of the instruction and ending with an instruction data indicating the end thereof is processed as a single instruction. As for an instruction of one instruction length unit,

a predecoded information indicating the head and end of the instruction is appended thereto.

Where a loading is carried out with units of every data block by a wrap-around, i.e., where the instruction head cannot be discriminated, a predecoded information indicating "NOT predecoded" state is appended to the instruction.

According to the cache memory 100a of the present embodiment, the instruction data from the instruction data inputting unit 120 and the corresponding predecoded result from the instruction predecoder 121a are stored via the I/O buffer 125a into the designated address of the data memory 126a in the cache "miss" state. On the other hand, where the cache "hit" state is encountered, the corresponding instruction data and the predecoded information are read out from the data memory 126a and fed via the I/O buffer 125a and the instruction data outputting unit 124a to the CPU 20. Therefore, the CPU 20 can utilize the predecoded information fed from the data memory 126a without any overhead and thus simultaneously decode a plurality of instructions.

Also, the instruction predecoder 121a checks a validity or invalidity of the predecoded information output from the data memory 126a and, where the predecoded information proves to be invalid, the I/O buffer 125a rewrites the invalid predecoded information to a right one based on the rewrite instruction from the instruction predecoder 121a. As a result, it is possible to remove an overhead in the next (second) instruction fetch and thereafter. Namely, although time for re-predecoding is required where the predecoded information from the data memory 126a proves to be invalid, the time is required only in the first instruction fetch. The time is negligible, compared with the case in the cache "miss" state (i.e., instruction fetch from the main memory 30).

Furthermore, even when a loading is carried out with units of every data block by a wrap-around in the registration of the instruction data into the data memory 126a, it is possible to usually store a valid predecoded information. As a result, it is possible to prevent a lowering in the performance which may possibly occur where a non-predecoded information is registered into the data memory 126a. This contributes to an improvement in the performance of the CPU 20.

Although the present invention has been disclosed and described by way of two embodiments, it is apparent to those skilled in the art that other embodiments and modifications of the present invention are possible without departing from the spirit or essential features thereof.

Claims

1. A cache memory (10) for storing at least one variable length instruction taken from a memory (30) and feeding the processed information to a control means (20), the cache memory comprising:
   an instruction length decoding means (11) for decoding an instruction length of the variable length instruction taken from the memory; and
   an instruction storing means (12) for storing
   the variable length instruction from the memory, together with an instruction length information obtained by the instruction length decoding means, wherein the variable length instruction and the instruction length information thereof are fed to the control means.

2. A cache memory as set forth in claim 1, wherein the instruction storing means comprises:
   a memory means (126) for storing an instruction fed from the memory and a corresponding instruction length information obtained by the instruction length decoding means; and
   means (125) for reading a corresponding instruction and an instruction length information thereof from the memory means in the cache hit state and transmitting the read data to the control means, and for receiving a corresponding instruction from the memory and an instruction length information thereof from the instruction length decoding means in the cache miss state and writing the received data into the memory means.

3. A cache memory as set forth in claim 2, further comprising means (123) for receiving a corresponding instruction from the memory and an instruction length information thereof from the instruction length decoding means in the cache miss state and transmitting the received data to the control means.

4. A cache memory as set forth in claim 2, wherein the instruction length decoding means comprises:
   means (211--214) for latching a variable length instruction data fed from the memory, with divided into a plurality of data blocks;
   means (221--224, 231--234) for decoding an instruction format and an effective address from the respective instruction data for very data block;
   means (241~244) for determining respective instruction lengths based on the decoded results for

every data block;

means (250) for generating an instruction length information for the latched variable length instruction data based on the respective determined instruction lengths, and generating a control signal for indicating which block of the plurality of data blocks is at the head of the variable length instruction; and

means (260), responsive to the control signal, for controlling the instruction length determining means to be enable or disable.

5. A cache memory as set forth in claim 4, further comprising a cache hit/miss judging means (116) for judging whether the cache memory encounters the cache hit state or the cache miss state.

6. A cache memory as set forth in claim 5, wherein the cache hit/miss judging means comprises means (117) for retaining a comparison address and means (118) for comparing an instruction address fed from the control means with the comparison address to thereby judge whether the cache memory encounters the cache hit state or the cache miss state.

7. A data processor comprising:

a processing means (20) for outputting an instruction address and fetching a variable length instruction; and

a cache memory (10,100) connected between the processing means and a main memory, the cache memory including:

an instruction length decoding means (11,121) for decoding an instruction length of the variable length instruction taken from the main memory; and

an instruction storing means (12,122) for storing the variable length instruction from the main memory, together with an instruction length information obtained by the instruction length decoding means, wherein the variable length instruction and the instruction length information thereof are fed to the processing means.

8. A data processor as set forth in claim 7, wherein the processing means comprises:

means (303) for storing a train of instruction codes (C) constituted by one set consisting of a basic part and an extended part or a combination of a plurality of sets;

means (315) for judging whether data of every unit length in the train of instruction codes is a basic part to thereby append a control mark (M) thereto;

means (316) for storing the control mark;

means (307) for designating a head position of a predetermined train of instruction codes to be selectively output from the code storing means;

means (317), responsive to outputs of the head position designating means and the mark storing means, for designating an output position of the predetermined train of instruction codes;

means (308), responsive to outputs of the output position designating means and the mark storing means, for selectively outputting the predetermined train of instruction codes from the code storing means;

means (309) for decoding instruction codes output from the selective outputting means;

means (310) for storing the predetermined train of instruction codes output from the selective outputting means;

means (312), responsive to outputs of the decoding means and the storing means, for generating an immediate value or a displacement value; and

means (313) for executing an instruction based on outputs of the generating means.

9. A data processor as set forth in claim 8, wherein the instruction storing means comprises:

a memory means (126) for storing an instruction fed from the memory and a corresponding instruction length information obtained by the instruction length decoding means; and

means (125) for reading a corresponding instruction and an instruction length information thereof from the memory means in the cache hit state and transmitting the read data to the control means, and for receiving a corresponding instruction from the memory and an instruction length information thereof from the instruction length decoding means in the cache miss state and writing the received data into the memory means.

10. A data processor as set forth in claim 9, further comprising means (123) for receiving a corresponding instruction from the memory and an instruction length information thereof from the instruction length decoding means in the cache miss state and transmitting the received data to the control means.

11. A data processor as set forth in claim 9, wherein the instruction length decoding means comprises
means (211~214) for latching a variable length instruction data fed from the memory, with divided into a plurality of data blocks;
means (221~224, 231~234) for decoding an instruction format and an effective address from the respective instruction data for every data block;
means (241~244) for determining respective instruction lengths based on the decoded results for every data block;
means (250) for generating an instruction length information for the latched variable length instruction data based on the respective determined instruction lengths, and generating a control signal for indicating which block of the plurality of data blocks is at the head of the variable length instruction; and
means (260), responsive to the control signal, for controlling the instruction length determining means to be enable or disable.

12. A data processor as set forth in claim 7, wherein the processing means comprises:
means (303) for storing a train of instruction codes (C) constituted by one set consisting of a basic part and an extended part or a combination of a plurality of sets;
means (315) for judging whether data of every unit length in the train of instruction codes is a basic part to thereby append a control mark (M) thereto;
means (316) for storing the control mark;
means (307a) for designating a head position of a predetermined train of instruction codes to be selectively output from the code storing means, by means of two read ports;
means (317), responsive to outputs of the head position designating means and the mark storing means, for designating an output position of the predetermined train of instruction codes;
a pair of output selecting units (308a,308b), responsive to outputs of the output position designating means and the mark storing means, each for selectively outputting the respective allocated portion of the predetermined train of instruction codes from the code storing means;
a pair of decoding units (309a,309b) each for decoding instruction codes output from the corresponding output selecting unit;
a pair of register units (310a,310b) each for storing instruction codes output from the corresponding output selecting unit; and
a pair of instruction executing units (313a,313b) each for executing the respective instruction based on outputs of the corresponding decoding unit and register unit.

13. A data processor as set forth in claim 12, wherein the instruction storing means comprises:
a memory means (126) for storing an instruction fed from the memory and a corresponding instruction length information obtained by the instruction length decoding means; and
means (125) for reading a corresponding instruction and an instruction length information thereof from the memory means in the cache hit state and transmitting the read data to the control means, and for receiving a corresponding instruction from the memory and an instruction length information thereof from the instruction length decoding means in the cache miss state and writing the received data into the memory means.

14. A data processor as set forth in claim 13, further comprising means (123) for receiving a corresponding instruction from the memory and an instruction length information thereof from the instruction length decoding means in the cache miss state and transmitting the received data to the control means.

15. A data processor as set forth in claim 13, wherein the instruction length decoding means comprises:
means (211~ 214) for latching a variable length instruction data fed from the memory, with divided into a plurality of data blocks;
means (221~224, 231~214) for decoding an instruction format and an effective address from the respective instruction data for every data block;
means (241~244) for determining respective instruction lengths based on the decoded results for every data block;
means (250) for generating an instruction length information for the latched variable length instruction data bas d on the respective determined instruction lengths, and gen rating a control signal for indicating which block of the plurality of data blocks is at the head of th variable length instruction; and
means (260), responsive to the control signal, for controlling the instruction length determining means to be enable or disable.

16. A cache memory (10a) for storing an instruction data (A) taken from a memory (30) and feeding the stored information to a control means (20), the cache memory comprising:
means (14) for generating a predecoded information (B) as an auxiliary of a decoding of the instruction data;
a memory means (15) for storing the predecoded information together with the instruction data;
means (16) for checking a predecoded information output from the memory means together with a corresponding instruction data in the cache hit state;
a rewriting means (17), where the checked predecoded information proves to be invalid or wrong, for rewriting the corresponding predecoded information in the memory means to a right one; and
a control information outputting means (18,18a, 18b), where the checked predecoded information proves to be invalid or wrong, for outputting a control information reflecting a result of the check to the control means.

17. A cache memory as set forth in claim 16, wherein the control information outputting means comprises means (18), where the checked predecoded information proves to be invalid or wrong, for informing the control means of the invalidity of the predecoded information.

18. A cache memory as set forth in claim 16, wherein the control information outputting means comprises means (18a), where the checked predecoded information proves to be invalid or wrong, for outputting a re-predecoded information to the control means.

19. A cache memory as set forth in claim 16, wherein the control information outputting means comprises means (18b), where the checked predecoded information proves to be invalid or wrong, for rewriting the corresponding predecoded information to be fed to the control means to a right one.

20. A cache memory as set forth in claim 16, wherein the memory means comprises:
a data memory (126a) for storing an instruction data fed from the memory and a corresponding predecoded information obtained by the predecoded information generating means; and
means (125a) for reading a corresponding instruction data and a predecoded information thereof from the data memory in the cache hit state and transmitting the read data to the control means, and for receiving a corresponding instruction data from the memory and a predecoded information thereof from the predecoded information generating means in the cache miss state and writing the received data into the data memory.

21. A cache memory (10b) for storing an instruction data (A) taken from a memory (30) and feeding the stored information to a control means (20), the cache memory comprising:
a memory means (15) for storing the instruction data together with a predecoded information (B) as an auxiliary of a decoding of the instruction data;
means (19) for generating the predecoded information and checking a predecoded information output from the memory means together with a corresponding instruction data in the cache hit state;
a rewriting means (17), where the checked predecoded information proves to be invalid or wrong, for rewriting the corresponding predecoded information in the memory means to a right one; and
a control information outputting means (18,18a, 18b), where the checked predecoded information proves to be invalid or wrong, for outputting a control information reflecting a result of the check to the control means.

22. A data processor comprising:
a processing means (20) for outputting an instruction address and fetching a variable length instruction; and
a cache memory (10a,100a) connected between the processing means and a main memory, the cache memory including:
means (14) for generating a predecoded information as an auxiliary of a decoding of the instruction data;
a memory means (15) for storing the predecoded information together with the instruction data;
means (16) for checking a predecoded information output from the memory means together with a corresponding instruction data in the cache hit state;
a rewriting means (17), where the checked predecoded information proves to be invalid or wrong, for rewriting the corresponding predecoded information in the memory means to a right one; and
a control information outputting means (18,18a,18b), where the checked predecoded information

proves to be invalid or wrong, for outputting a control information reflecting a result of the check to the processing means.

Fig. 1

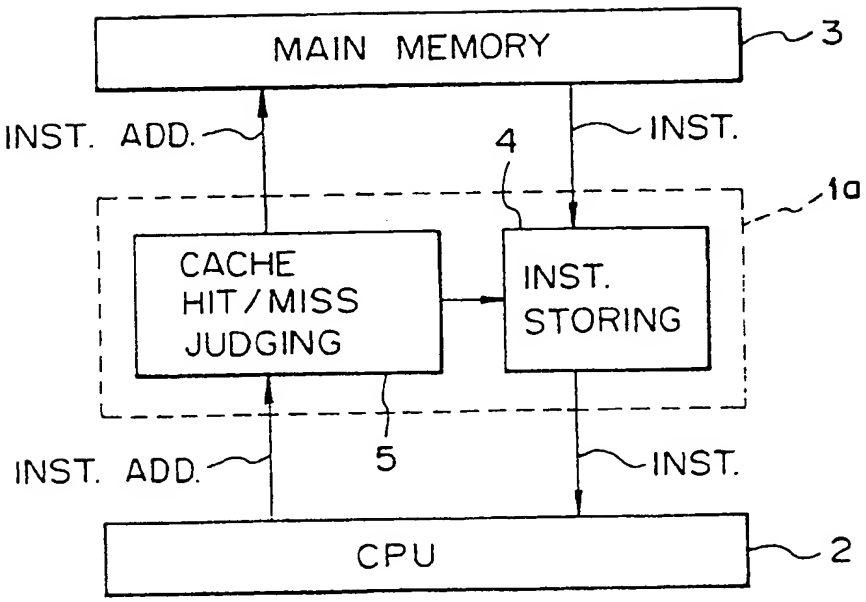

MAIN MEMORY
3
INST. ADD.
4
INST.
1a
CACHE HIT/MISS JUDGING
INST. STORING
INST. ADD.
5
INST.
CPU
2

Fig. 2

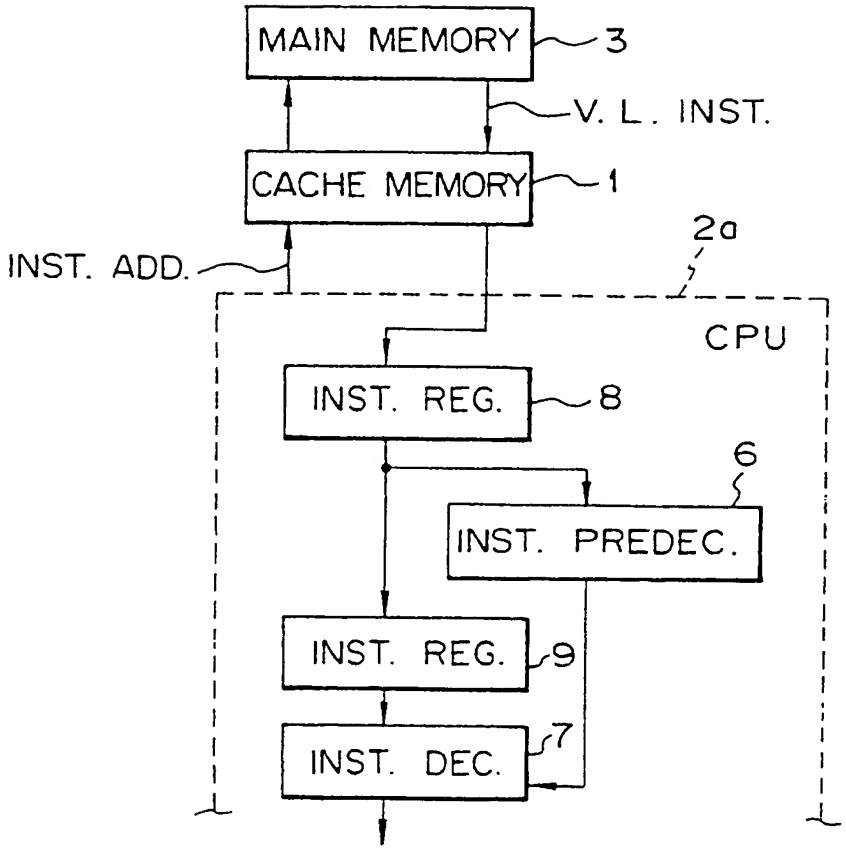

MAIN MEMORY
3
V. L. INST.
CACHE MEMORY
1
2a
INST. ADD.
CPU
INST. REG.
8
6
INST. PREDEC.
INST. REG.
9
INST. DEC.
7

Fig. 3

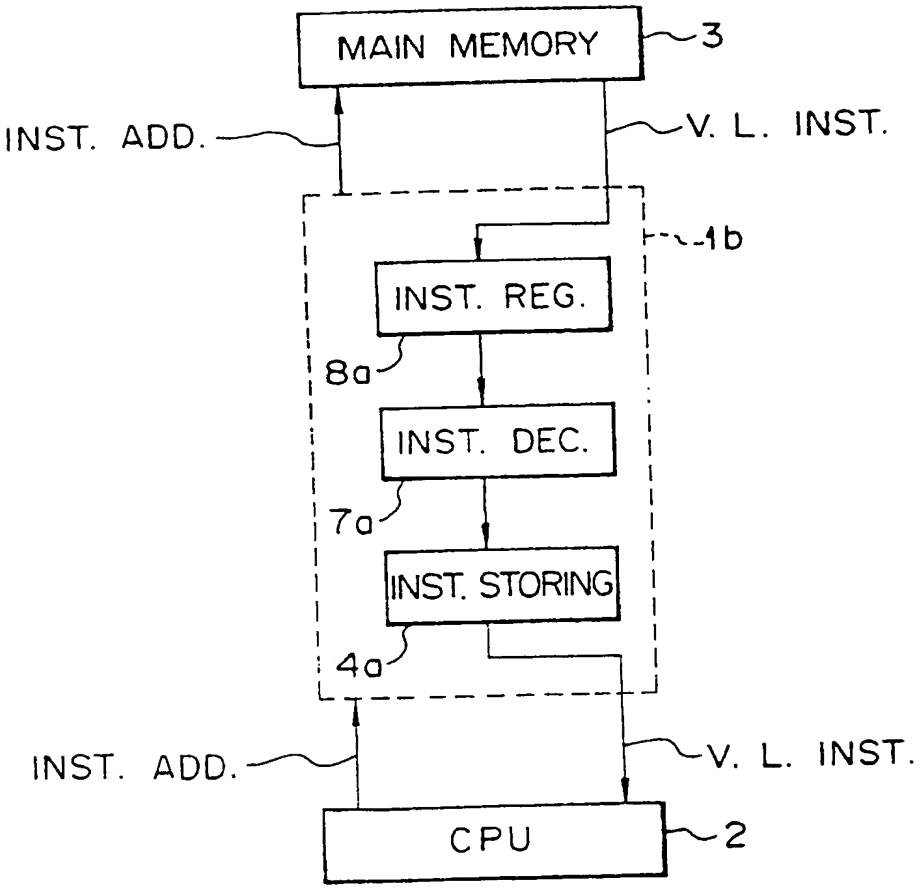

MAIN MEMORY
3
INST. ADD.
V. L. INST.
1b
INST. REG.
8a
INST. DEC.
7a
INST. STORING
4a
INST. ADD.
V. L. INST.
CPU
2

Fig. 4

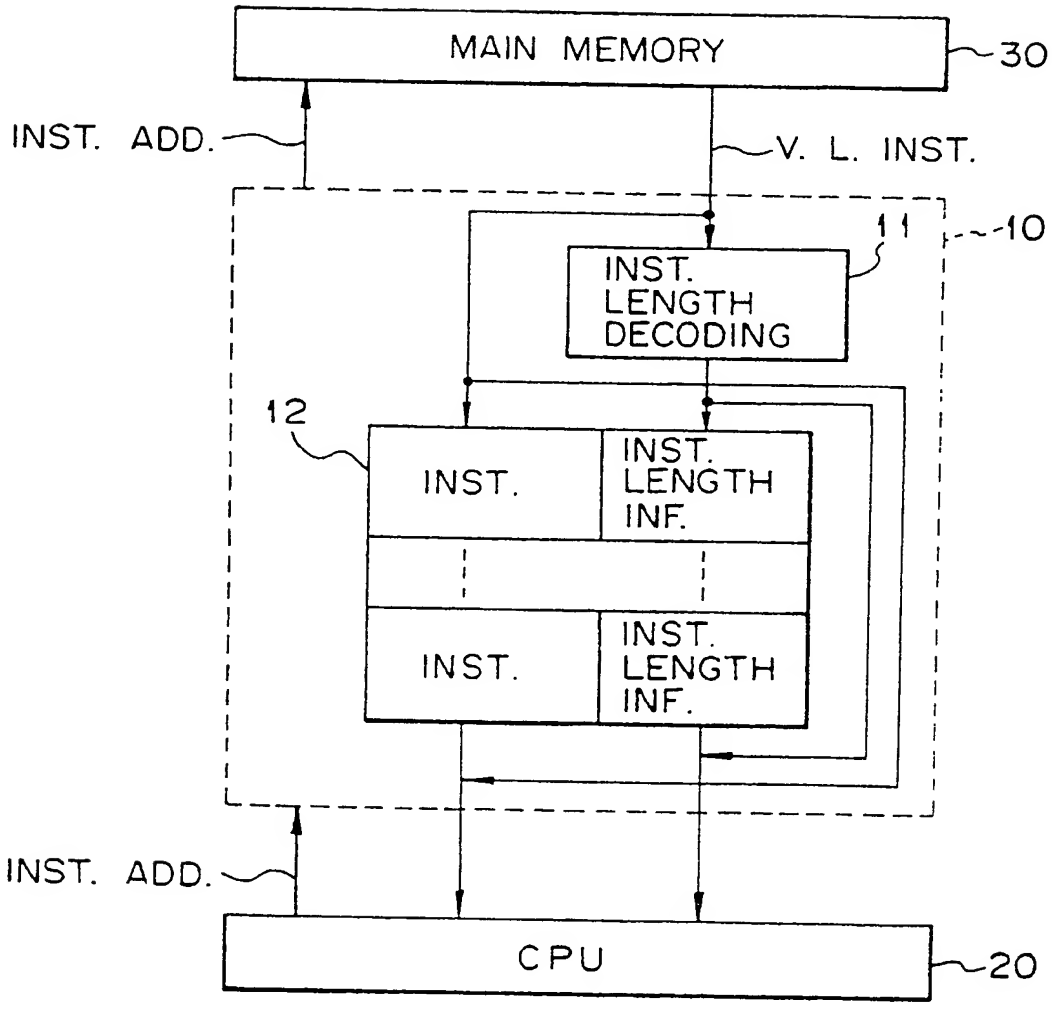

MAIN MEMORY
30
INST. ADD.
V. L. INST.
11
10
INST.
LENGTH
DECODING
12
INST.
INST.
LENGTH
INF.
INST.
INST.
LENGTH
INF.
INST. ADD.
CPU
20

Fig. 5

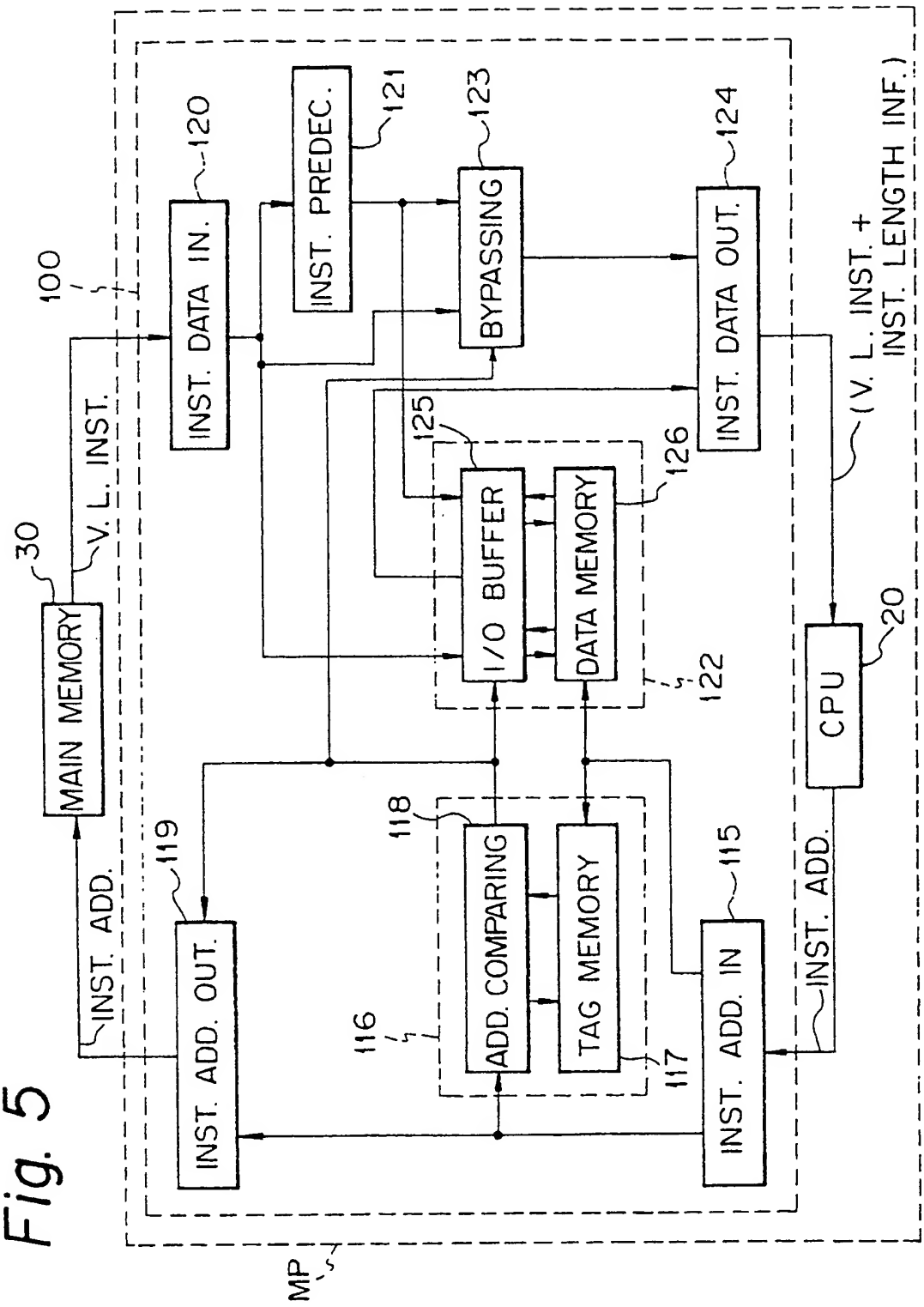

MP
100
MAIN MEMORY
30
V. L. INST.
INST. ADD.
INST. ADD. OUT.
119
INST. DATA IN.
120
INST. PREDEC.
121
BYPASSING
123
INST. DATA OUT.
124
I/O BUFFER
125
DATA MEMORY
126
122
ADD. COMPARING
118
TAG MEMORY
116
117
INST. ADD. IN
115
CPU
20
INST. ADD.
(V. L. INST. + INST. LENGTH INF.)

Fig. 6

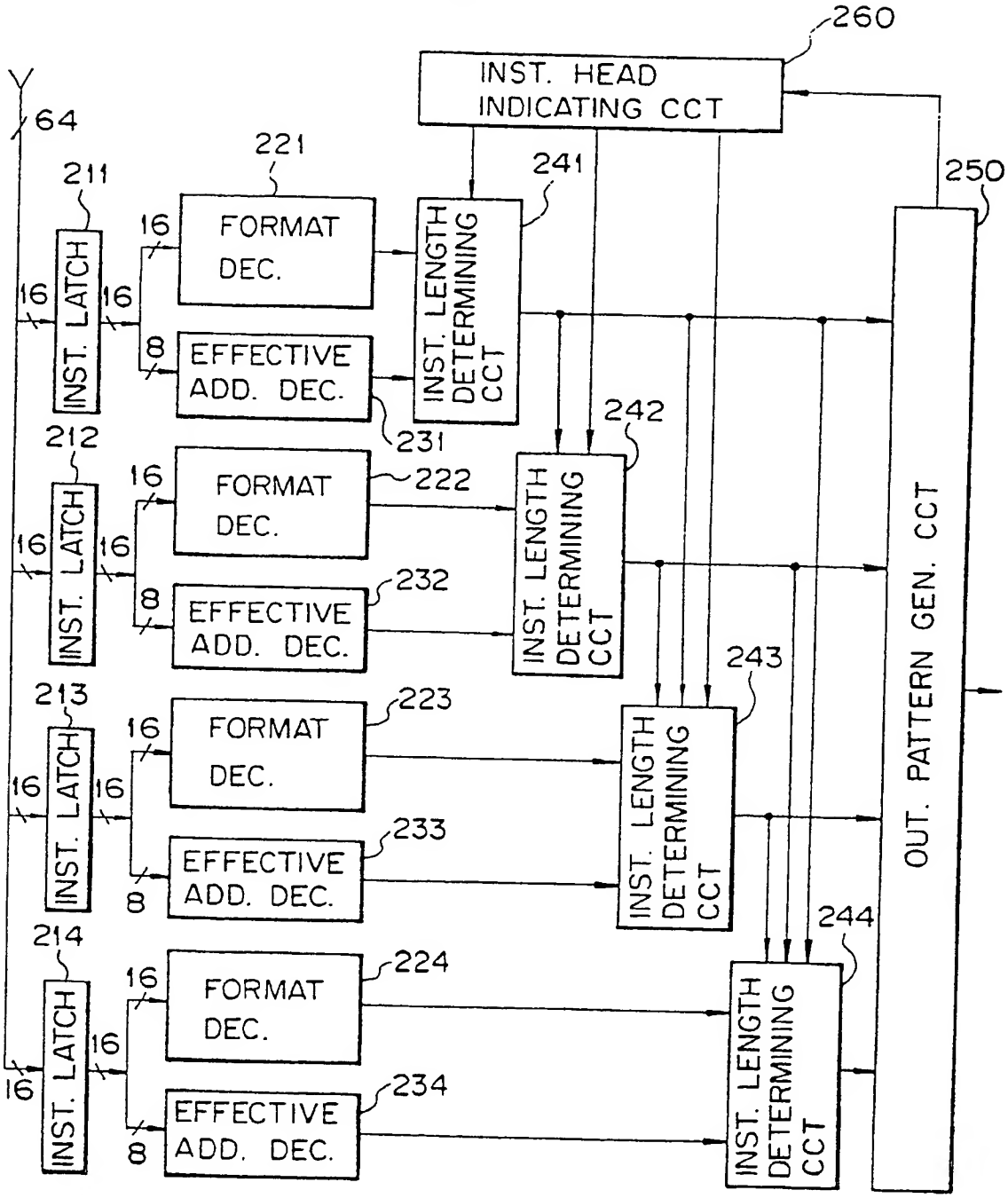

260
INST. HEAD INDICATING CCT
64
221
241
250
211
16
FORMAT DEC.
INST. LENGTH DETERMINING CCT
INST. LATCH
16
16
8
EFFECTIVE ADD. DEC.
231
242
212
222
232
213
223
243
233
214
224
244
234
OUT. PATTERN GEN. CCT

Fig. 7

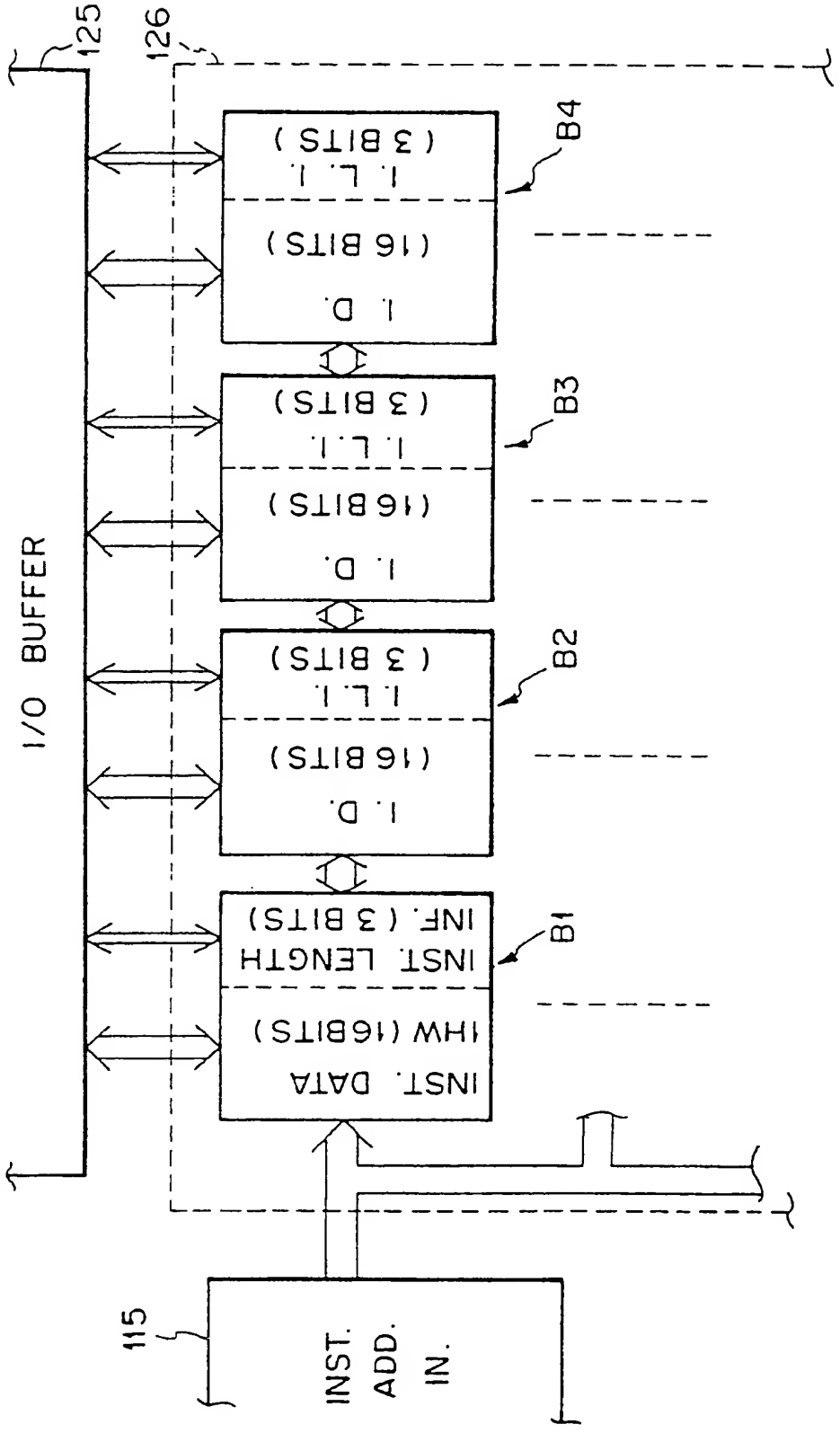

125
126
I/O BUFFER
B4
B3
B2
B1
I. L. I. (3 BITS)
I. D. (16 BITS)
INF. (3 BITS)
INST. LENGTH
INST. DATA
IHW (16BITS)
115
INST. ADD. IN.

Fig. 8

Fig. 8A	Fig. 8B

Fig. 8A

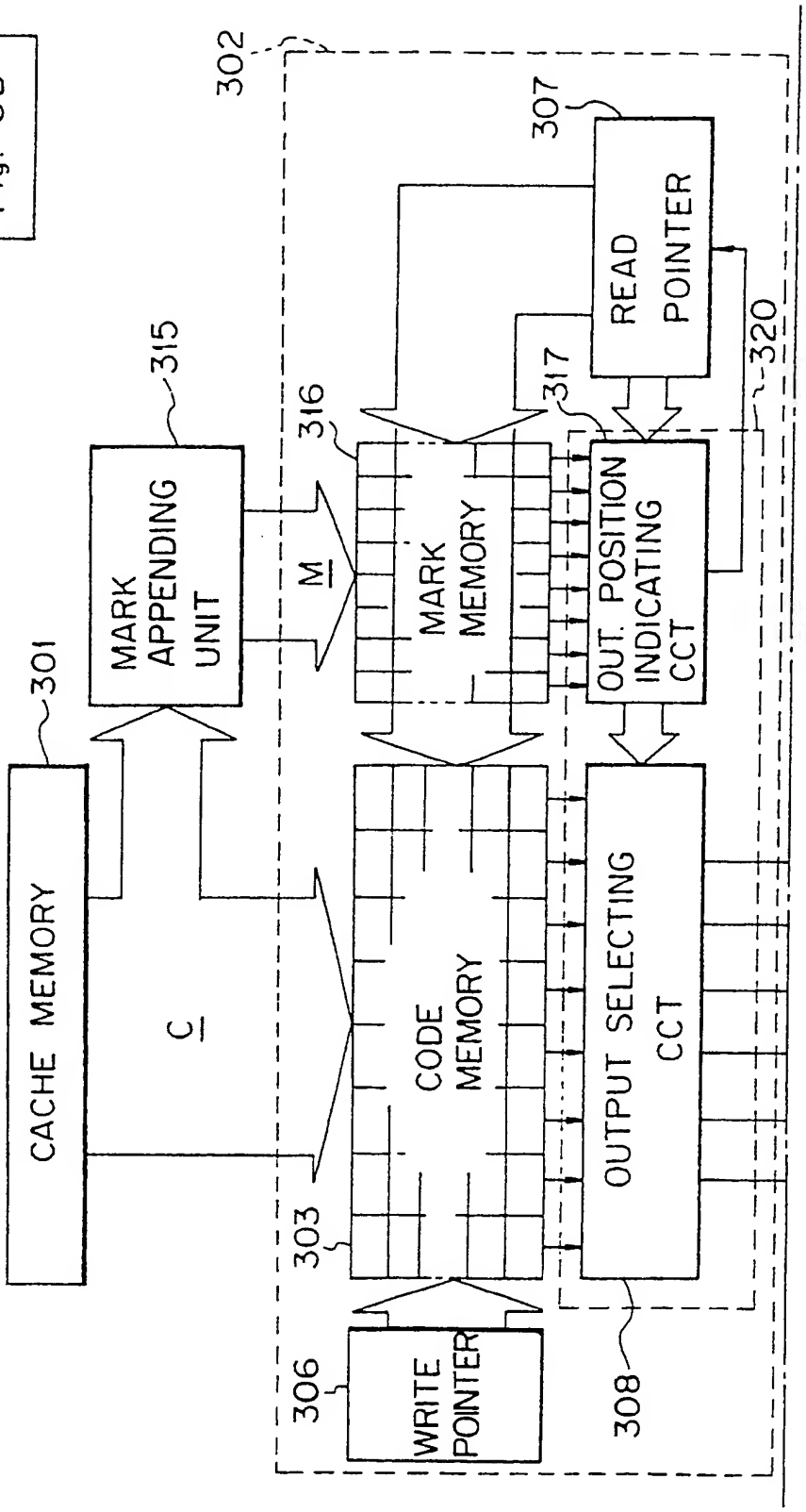


Fig. 8B

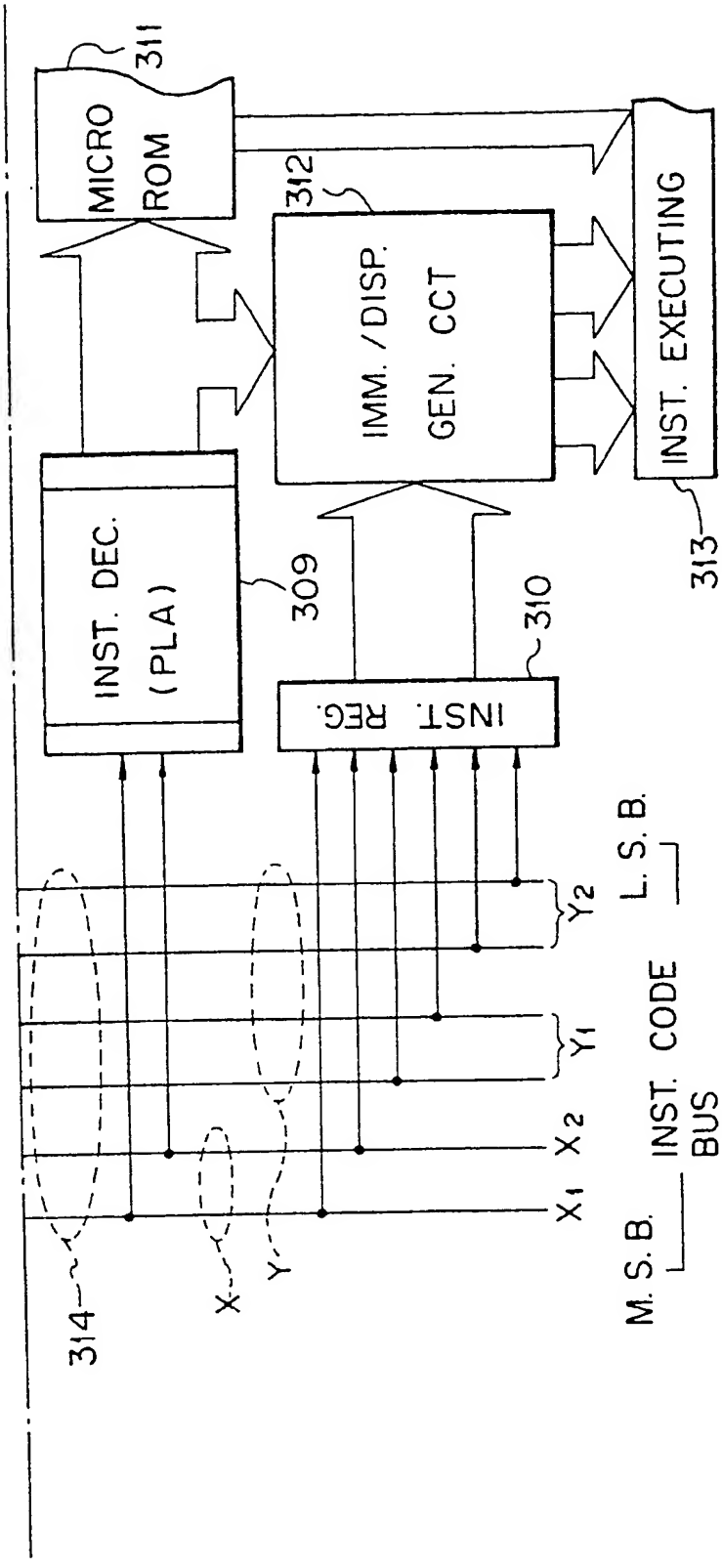

MICRO ROM
311
312
IMM. / DISP. GEN. CCT
INST. EXECUTING
313
INST. DEC. (PLA)
309
310
INST. REG.
L. S. B.
Y2
Y1
X2
X1
INST. CODE BUS
M. S. B.
314
X
Y

Fig. 9

Fig. 9A	Fig. 9B
	Fig. 9C

Fig. 9A

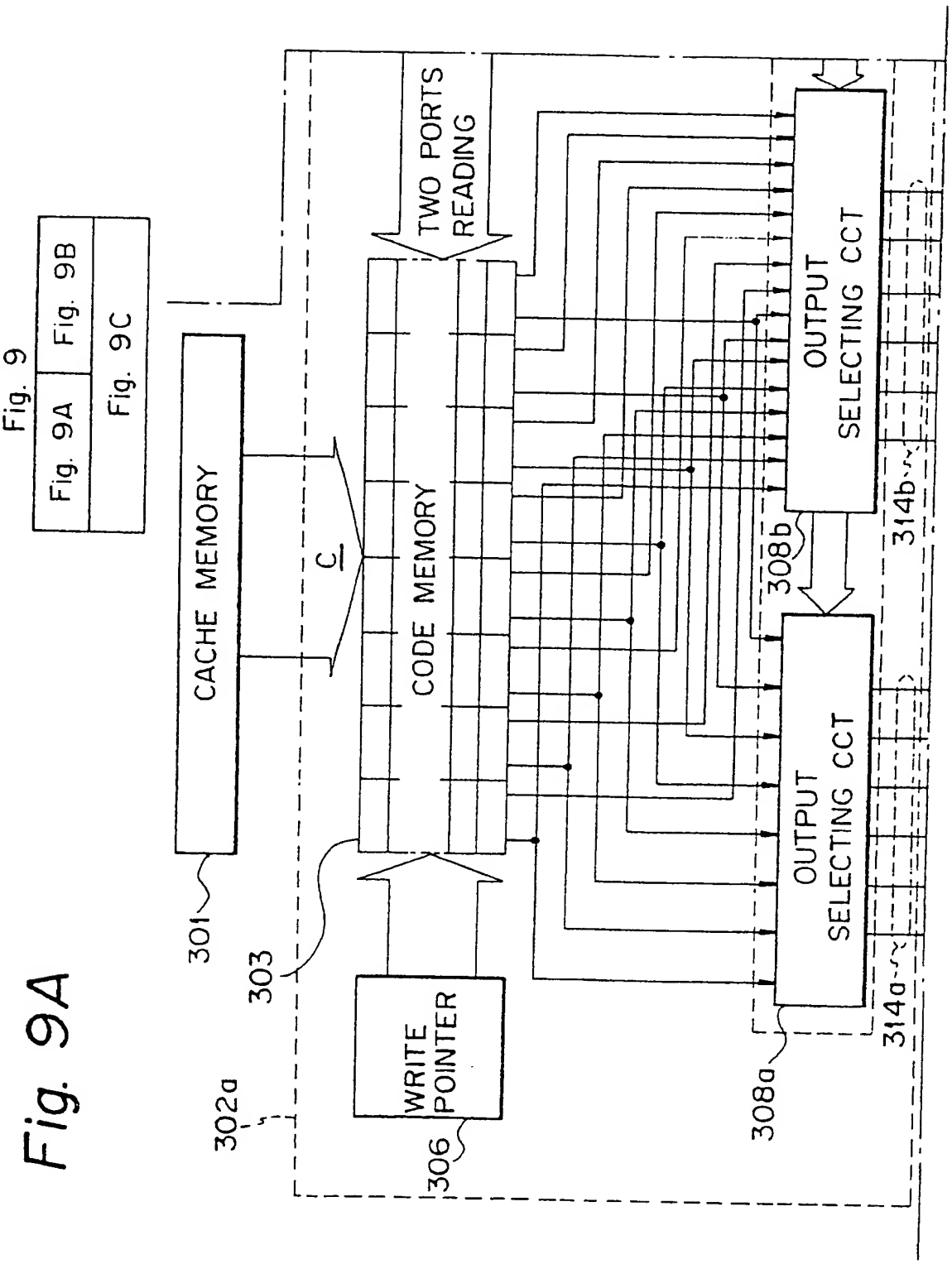


Fig. 9B

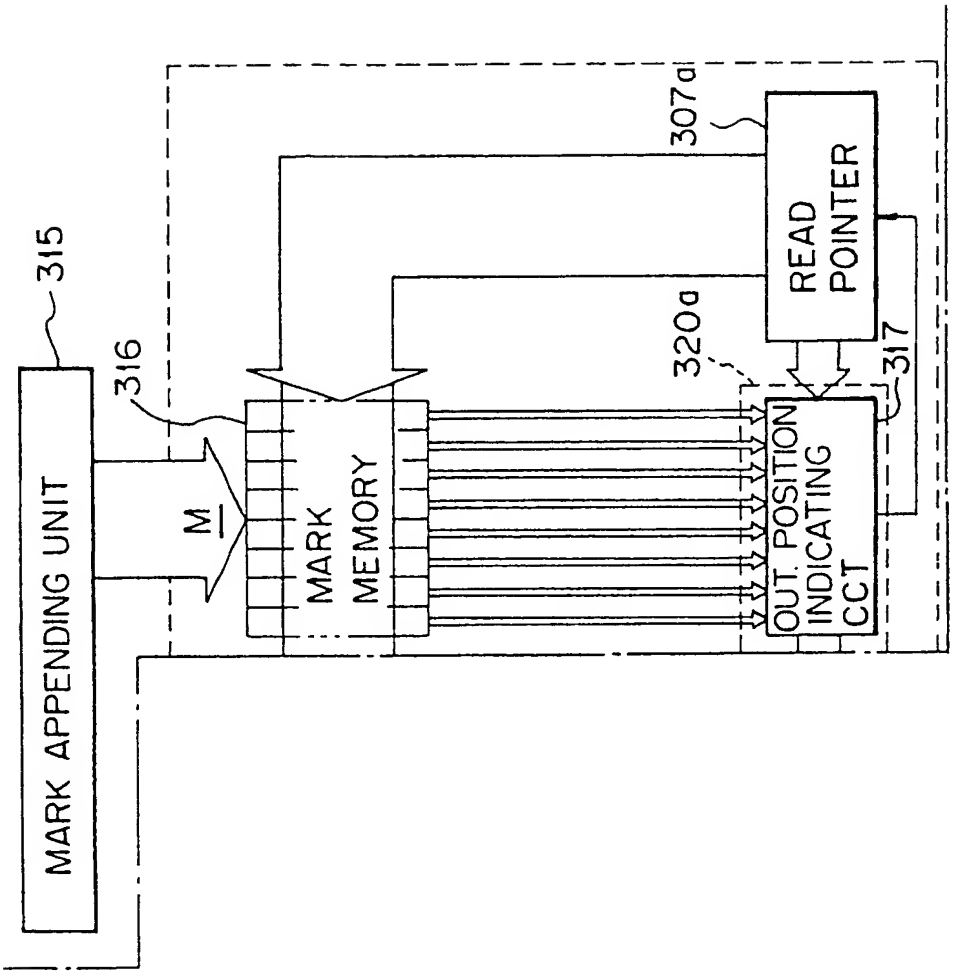

MARK APPENDING UNIT
315
316
M
MARK MEMORY
307a
READ POINTER
320a
317
OUT. POSITION INDICATING CCT

Fig. 9C

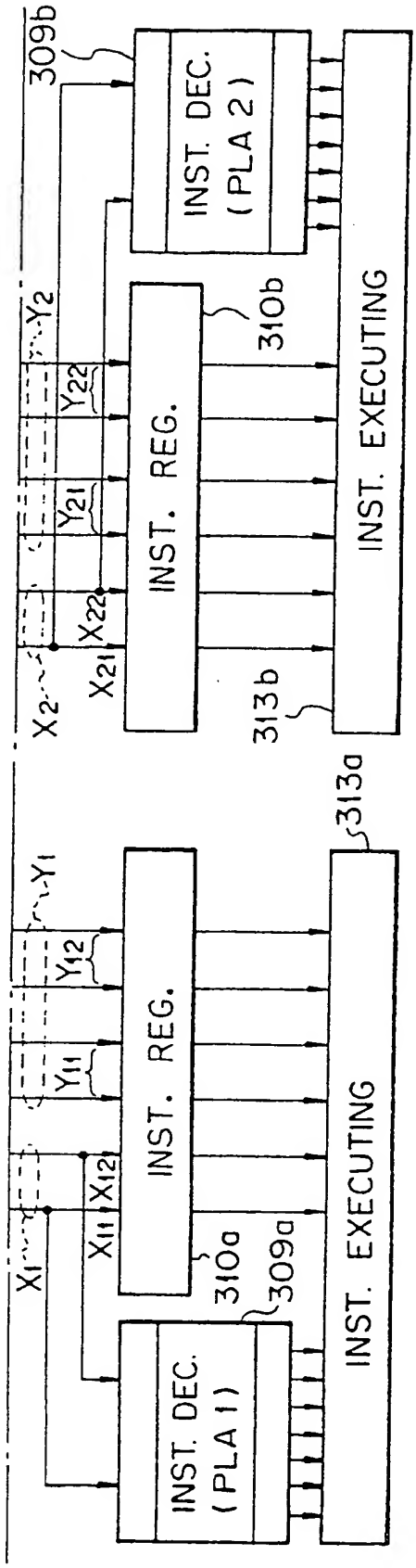

INST. DEC. (PLA 1)
INST. REG.
INST. EXECUTING
INST. DEC. (PLA 2)
INST. REG.
INST. EXECUTING
X1
X11
X12
Y1
Y11
Y12
X2
X21
X22
Y2
Y21
Y22
309a
310a
313a
309b
310b
313b

Fig. 10 PRIOR ART

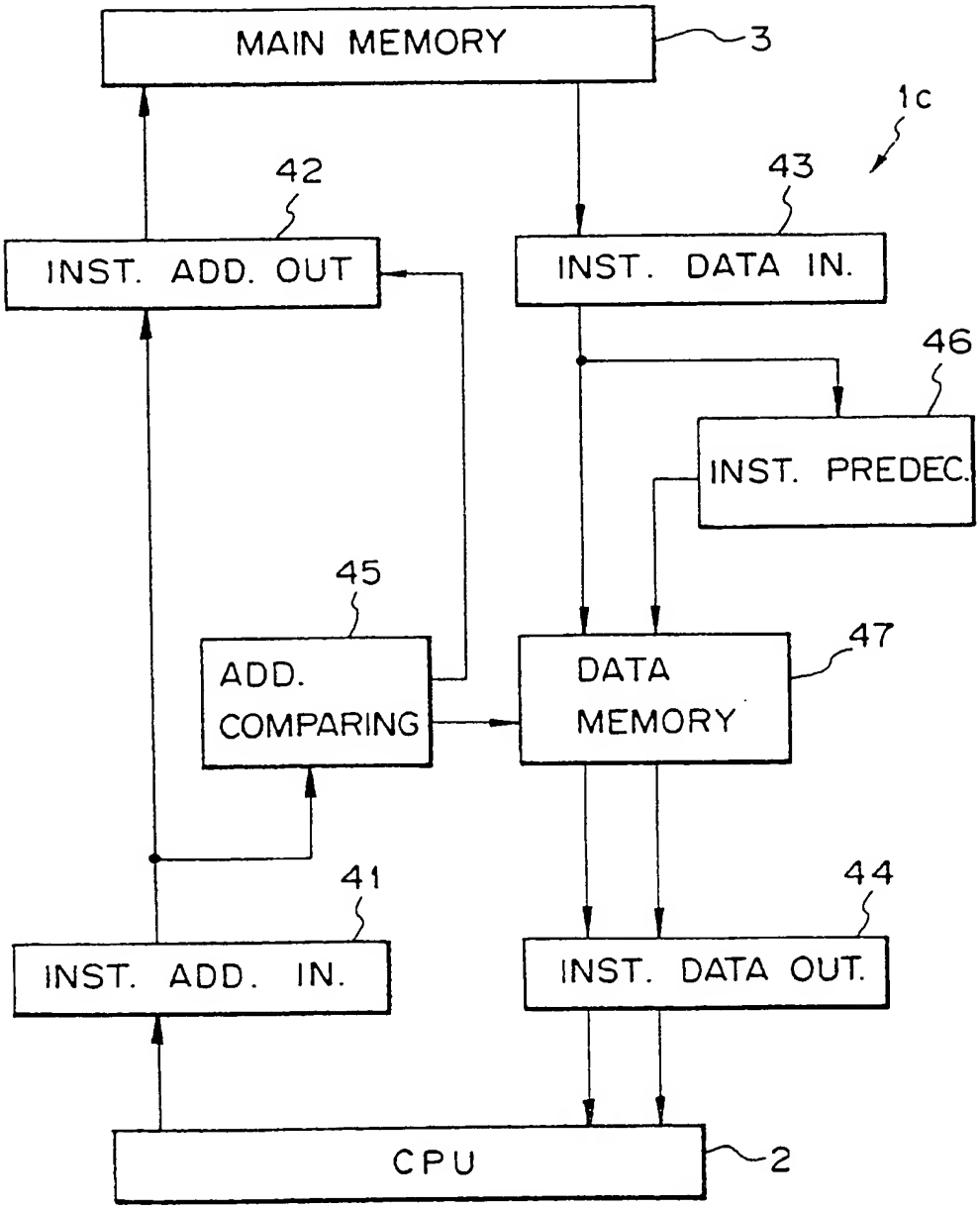
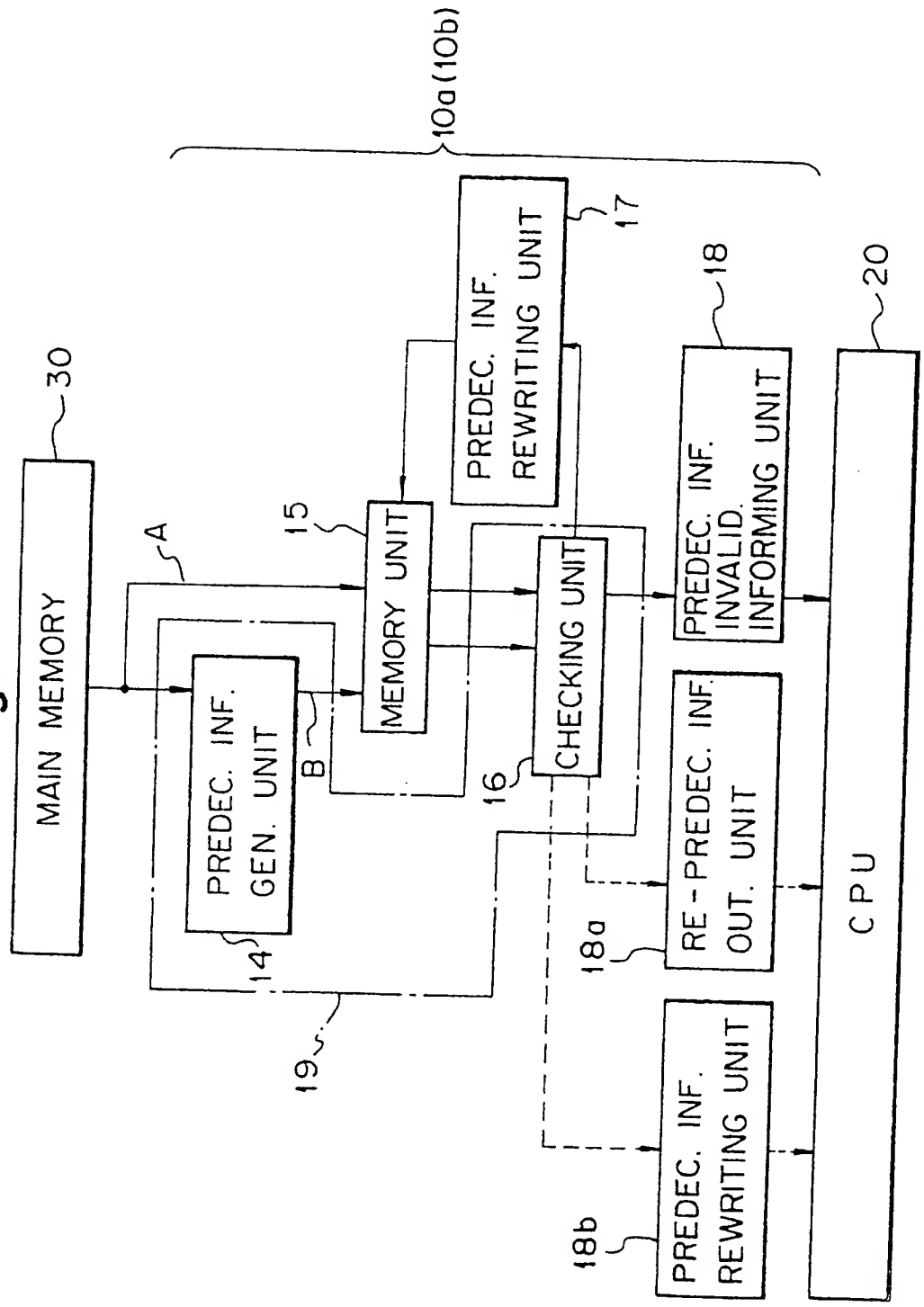


Fig. 11


10a (10b)
MAIN MEMORY
30
A
PREDEC. INF. GEN. UNIT
14
B
15
MEMORY UNIT
PREDEC. INF. REWRITING UNIT
17
CHECKING UNIT
16
PREDEC. INF. INVALID. INFORMING UNIT
18
20
CPU
RE-PREDEC. INF. OUT. UNIT
18a
PREDEC. INF. REWRITING UNIT
18b
19

Docket # P2001,0304
Applic. #
Applicant: Heino Hartlichedal.
Lerner and Greenberg, P.A.
Post Office Box 2480
Hollywood, FL 33022-2480
Tel: (954) 925-1100 Fax: (954) 925-1101

Fig. 12